/dev/world

2018

SIL for First Time Learners

Yusuke Kita

Mecari

@kitasuke

AUC

In Partnership With

RMIT UNIVERSITY

Hi, I'm Yusuke

@kitasuke

Why SIL?

Why SIL?

Why SIL?

→ Better idea of Swift type system 🍎

Why SIL?

→ Better idea of Swift type system 🍎

→ Optimizations magic 🔮✨

Why SIL?

→ Better idea of Swift type system 🍎

→ Optimizations magic 🔮✨

→ For fun and profit 😎

SIL

Swift Intermediate Language

SIL for First Time Learners, Yusuke Kita (@kitasuke), /dev/world/2018

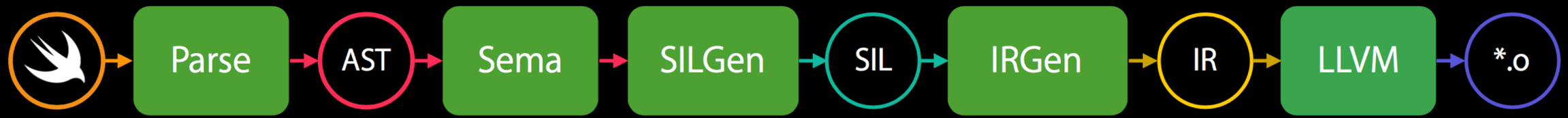
SIL is a language specific Intermediate Representation

Swift

SIL for First Time Learners, Yusuke Kita (@kitasuke), /dev/world/2018

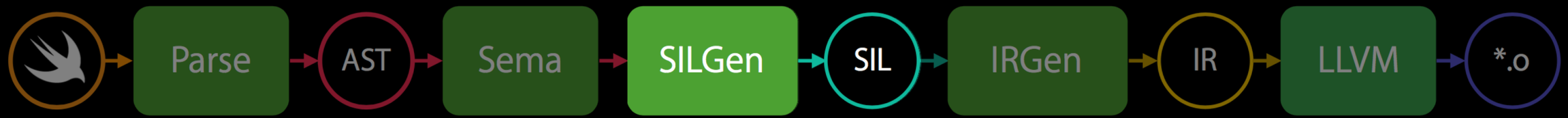
Swift Compiler

Swift Compiler



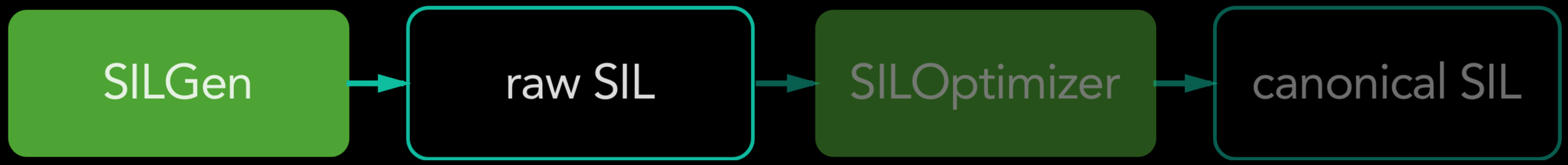
SIL for First Time Learners, Yusuke Kita (@kitasuke), /dev/world/2018

Swift Compiler

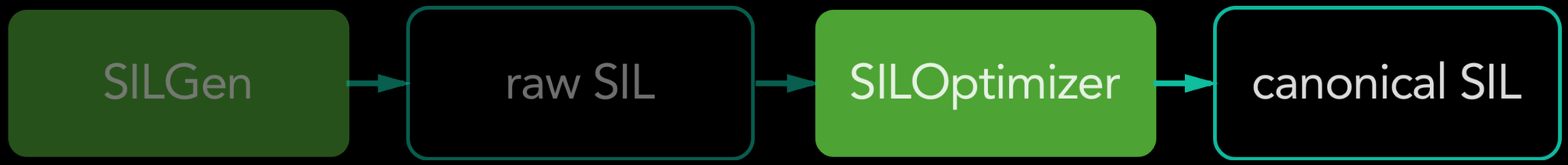


SIL for First Time Learners, Yusuke Kita (@kitasuke), /dev/world/2018

raw SIL



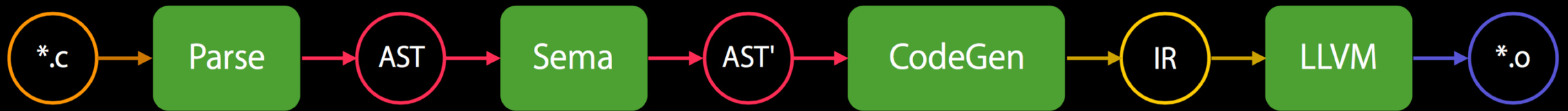
canonical SIL



Objective-C

Clang

Clang



SIL

How *SIL* works?

sample.swift

```
func number() -> Int {
    let x: Int
    x = 1
    return x
}
```

Emit SIL

```
$swiftc -emit-sil sample.swift > sample.sil
```

sample.sil

```
sil hidden @_T06sample6numberSiyF : $@convention(thin) () -> Int {
bb0:
    %0 = alloc_stack $Int, let, name "x"
    %1 = integer_literal $Builtin.Int64, 1
    %2 = struct $Int (%1 : $Builtin.Int64)
    store %2 to %0 : $*Int
    dealloc_stack %0 : $*Int
    return %2 : $Int
}
```

sample.sil

```
sil hidden @_T06sample6numberSiyF : $@convention(thin) () -> Int {
bb0:
    %0 = alloc_stack $Int, let, name "x"
    %1 = integer_literal $Builtin.Int64, 1
    %2 = struct $Int (%1 : $Builtin.Int64)
    store %2 to %0 : $*Int
    dealloc_stack %0 : $*Int
    return %2 : $Int
}
```

sample.sil

```
sil hidden @_T06sample6numberSiyF : $@convention(thin) () -> Int {
bb0:
    %0 = alloc_stack $Int, let, name "x"
    %1 = integer_literal $Builtin.Int64, 1
    %2 = struct $Int (%1 : $Builtin.Int64)
    store %2 to %0 : $*Int
    dealloc_stack %0 : $*Int
    return %2 : $Int
}
```

sample.sil

```
sil hidden @_T06sample6numberSiyF : $@convention(thin) () -> Int {
bb0:
    %0 = alloc_stack $Int, let, name "x"
    %1 = integer_literal $Builtin.Int64, 1
    %2 = struct $Int (%1 : $Builtin.Int64)
    store %2 to %0 : $*Int
    dealloc_stack %0 : $*Int
    return %2 : $Int
}
```

sample.sil

```
sil hidden @_T06sample6numberSiyF : $@convention(thin) () -> Int {
bb0:
    %0 = alloc_stack $Int, let, name "x"
    %1 = integer_literal $Builtin.Int64, 1
    %2 = struct $Int (%1 : $Builtin.Int64)
    store %2 to %0 : $*Int
    dealloc_stack %0 : $*Int
    return %2 : $Int
}
```

sample.sil

```
sil hidden @_T06sample6numberSiyF : $@convention(thin) () -> Int {
bb0:
    %0 = alloc_stack $Int, let, name "x"
    %1 = integer_literal $Builtin.Int64, 1
    %2 = struct $Int (%1 : $Builtin.Int64)
    store %2 to %0 : $*Int
    dealloc_stack %0 : $*Int
    return %2 : $Int
}
```

sample.sil

```
sil hidden @_T06sample6numberSiyF : $@convention(thin) () -> Int {
bb0:
    %0 = alloc_stack $Int, let, name "x"
    %1 = integer_literal $Builtin.Int64, 1
    %2 = struct $Int (%1 : $Builtin.Int64)
    store %2 to %0 : $*Int
    dealloc_stack %0 : $*Int
    return %2 : $Int
}
```

sample.sil

```
sil hidden @_T06sample6numberSiyF : $@convention(thin) () -> Int {
bb0:
    %0 = alloc_stack $Int, let, name "x"
    %1 = integer_literal $Builtin.Int64, 1
    %2 = struct $Int (%1 : $Builtin.Int64)
    store %2 to %0 : $*Int
    dealloc_stack %0 : $*Int
    return %2 : $Int
}
```

sample.sil

```
sil hidden @_T06sample6numberSiyF : $@convention(thin) () -> Int {
bb0:
    %0 = alloc_stack $Int, let, name "x"
    %1 = integer_literal $Builtin.Int64, 1
    %2 = struct $Int (%1 : $Builtin.Int64)
    store %2 to %0 : $*Int
    dealloc_stack %0 : $*Int
    return %2 : $Int
}
```

Emit SIL with optimizations

```
$swiftc -emit-sil -O sample.swift > sample.sil
```

`sample.sil` with optimizations

```
sil hidden @_T06sample6numberSiyF : $@convention(thin) () -> Int {
bb0:
    %0 = integer_literal $Builtin.Int64, 1
    %1 = struct $Int (%0 : $Builtin.Int64)
    return %1 : $Int
}
```

`sample.sil` with optimizations

```
sil hidden @_T06sample6numberSiyF : $@convention(thin) () -> Int {
bb0:
    %0 = integer_literal $Builtin.Int64, 1
    %1 = struct $Int (%0 : $Builtin.Int64)
    return %1 : $Int
}
```

`sample.sil` with optimizations

```
sil hidden @_T06sample6numberSiyF : $@convention(thin) () -> Int {
bb0:
    %0 = integer_literal $Builtin.Int64, 1
    %1 = struct $Int (%0 : $Builtin.Int64)
    return %1 : $Int
}
```

`sample.sil` with optimizations

```
sil hidden @_T06sample6numberSiyF : $@convention(thin) () -> Int {
bb0:
    %0 = integer_literal $Builtin.Int64, 1
    %1 = struct $Int (%0 : $Builtin.Int64)
    return %1 : $Int
}
```

`sample.sil` with optimizations

```
sil hidden @_T06sample6numberSiyF : $@convention(thin) () -> Int {
bb0:
    %0 = integer_literal $Builtin.Int64, 1
    %1 = struct $Int (%0 : $Builtin.Int64)
    return %1 : $Int
}
```

`sample.sil` with optimizations

```
sil hidden @_T06sample6numberSiyF : $@convention(thin) () -> Int {
bb0:
    %0 = integer_literal $Builtin.Int64, 1
    %1 = struct $Int (%0 : $Builtin.Int64)
    return %1 : $Int
}
```

Diff

```
sil hidden @_T06sample6numberSiyF : $@convention(thin) () -> Int {
bb0:
    %0 = alloc_stack $Int, let, name "x"
    %1 = integer_literal $Builtin.Int64, 1
    %2 = struct $Int (%1 : $Builtin.Int64)
    store %2 to %0 : $*Int
    dealloc_stack %0 : $*Int
    return %2 : $Int
}
```

sample.swift

```
func number() -> Int {
    let x: Int
    x = 1
    return x
}
```

`sample.swift` with optimizations

```
func number() -> Int {
    return 1
}
```

Mem2Reg

Register Promotion of Stack Allocations

Optimizations 🔮✨

Optimization flags

Optimization flags

→ –Onone

Optimization flags

→ –Onone

→ –O

Optimization flags

→ -Onone

→ -O

→ -Ounchecked

Optimization flags

→ `-Onone`

→ `-O`

→ `-Ounchecked`

→ `-Osize`

Tips for debug

→ `-Xllvm -sil-print-all`

→ `-Xllvm -sil-print-only-functions`

→ `-Xllvm -sil-print-before/after/around`

Summary

Summary

→ Optimize, optimize and optimize 💪

Summary

→ Optimize, optimize and optimize 💪

→ Better idea of how Swift Compiler works 💯

Summary

→ Optimize, optimize and optimize 💪

→ Better idea of how Swift Compiler works 💯

→ Definitely worth learning 🏆

References

→ Swift type in SIL

→ swift/docs/SIL.rst

→ Debugging the Swift Compiler

→ Swift's High-Level IR: A Case Study of Complementing LLVM IR with Language-Specific Optimization

Thank you!